arduino beginner

IDE und einfaches Programmieren :-)

engelbert gruber

19.11.2020
Rev. 0.2

INTRODUCTION

Was macht die IDE was kann ein arduino.

Was kann man dem Gerät anschaffen = programmieren

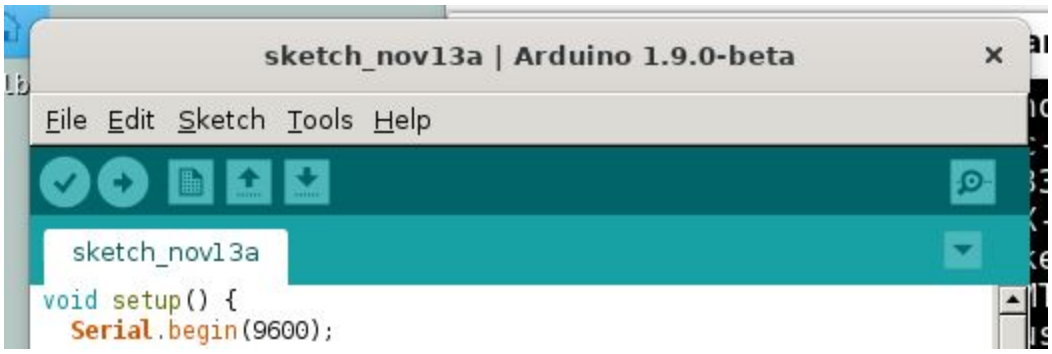
MATERIALS

1. arduino
2. PC mit arduino-IDE
3. Interesse

DIE IDE

Eine IDE, Integrated Development Environment, eine integrierte Entwicklungsumgebung verbindet Editor, Compiler, Linker. Im Falle des arduino handelt es sich um eine Cross Compiler-Umgebung, da die erstellten Programme für ein anderes System gemacht werden. Dazu ist auch das Programm zur Übertragung des erstellten arduino-Programms auf die arduino-Hardware enthalten.

Die arduino-IDE ist sehr einfach gehalten (die Komplexität steckt unter der GUI, weil verschiedene Hardwareplattformen unterstützt werden z.B. ATMEL, ARM, ESP).



Das Menu enthält nur fünf Einträge, und es gibt fünf Buttons.

Die Menueinträge

Nur die wichtigen Aktionen.

- File - New, Open und Save sind auch über Buttons erreichbar oder noch einfacher Strg+N, Strg+O, Strg+S.
- File - Open Recent enthält die zuletzt geöffneten Dateien, falls man wie ich nicht immer weiß was man zuletzt gemacht hat.
- File - Examples enthält viele Beispielprogramme.

Tools

- Tools - Port : Hier wählt man die Schnittstelle an der der arduino angesteckt ist.
- Tools - Board : Hier wählt man den Typ des angeschlossenen arduino.

Sketch

- Verify/Compile - oder der Button mit dem Hacken - oder Strg+R, kompiliert die geöffnete Datei.
- Upload - oder der Button mit dem Pfeil nach rechts - oder Strg+U, überträgt das Programm zum arduino ... wenn in Tools Port und Board richtig eingestellt sind.

PROGRAMME

Ein arduino-Programm besteht aus mindestens zwei Funktionen, **setup** und **loop**.

setup wird beim Programmstart einmal aufgerufen.

Die **loop**-Funktion wird immer wieder aufgerufen.

Die Programmiersprache ist C++ (denke ich) damit schaut ein Minimalprogramm so aus

```
void setup() {
}
void loop() {
}
```

Blinken

Am arduino ist eine LED verbaut die wir in unseren Programmen verwenden können.

Die LED ist am Anschluss “LED_BUILTIN” angeschlossen und dieser muss als “OUTPUT” konfiguriert werden.

```
void setup() {
pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
digitalWrite(LED_BUILTIN, HIGH);
}
```

Man kann den ganzen Code ohne Einrückung eintippen und dann via Menu “Tools - Auto Format” oder Strg+T drücken die Zeilen Einrückung korrigieren. Man kann auch die geschwungenen Klammer am Zeilenende setzen, beim Autoformat werden sie auf eine neue Zeile verschoben. Danach sieht der Code so aus.

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
}
```

Das Einfärben des Programmcodes ist auch hilfreich, wenn man statt “`LED_BUILTIN`” “`LED_BUILDIN`” ändert sich die Farbe auf Schwarz. **Aber:** “Serial.readline” ist orange gibt es aber nicht.

Nach dem Upload des Programms auf den arduino ist die/eine gelbe LED eingeschalten.

Zum Test ob das auch wirklich unser Programm ist können wir entweder das Programm

ändern und statt “HIGH” “LOW” schreiben oder beides

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  digitalWrite(LED_BUILTIN, LOW);
}
```

Mit diesem Programm wird die LED nur kurz eingeschalten und leuchtet nur sehr schwach. Am Oszilloskop sieht das so aus

5V Amplitude 2,6µs HIGH und 2,8µs LOW.

Zum Verlangsamen, damit wir das Blinken sehen verwenden wir “delay”.

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(500);
```

```
  digitalWrite(LED_BUILTIN, LOW);
  delay(500);
}
```

Die “500” sind die Zeit in Millisekunden die das Programm … pausiert.

Aber wie findet man den Pin an dem die LED angeschlossen ist ?

RTFM … einfach weiterlesen

Kommunikation mit dem arduino-Programm

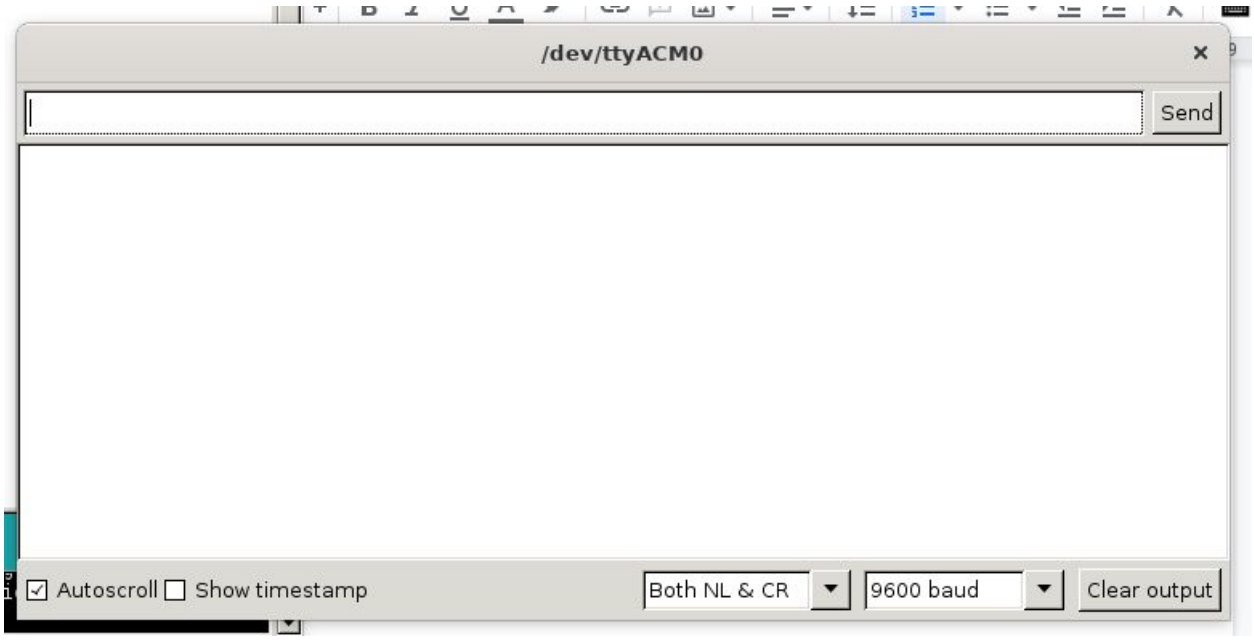
Der arduino hat kein Display und es ist nur eine LED verbaut um Messwerte kontrollieren zu können kann man diese über die USB-Serielle zum Computer schicken.

Die Serielle Schnittstelle muss am arduino initialisert werden.

```
void setup() {
  Serial.begin(9600);
}
```

“9600” ist die Baudrate, eine sehr langsame, mit der die Daten zwischen arduino und PC übertragen werden.

Via Menu “Tools - Serial Monitor” oder Strg+Shift+M öffnet ein Fenster in dem vom arduino geschickte Daten angezeigt werden.



Im unteren Fensterrand sieht man die Einstellung “9600 baud”. Wenn dieser Wert nicht zu dem im arduino eingestellten passt … ist die Anzeige mehr oder weniger ein Zufallsprodukt (= eine andere Übung). Damit können wir manche Sachen nachschauen, z.B. was ist LED_BUILTIN (eine Zahl) , welchen Wert hat sie.

```
void setup() {
  Serial.begin(9600);
  Serial.println(LED_BUILTIN);
}
```

Beim Compilieren des obigen Programms bekommt man die Fehlermeldung

Arduino: 1.8.13 (Linux), Board: "Arduino Uno"

```
/tmp/ccSsstsB.ltrans0.ltrans.o: In function `main':
/home/engelbert/Downloads/arduino-1.8.13/hardware/arduino/avr/cores/arduino/mai
n.cpp:46: undefined reference to `loop'
collect2: error: ld returned 1 exit status
exit status 1
Error compiling for board Arduino Uno.

This report would have more information with
"Show verbose output during compilation"
option enabled in File -> Preferences.
```

Der Fehler ist “`undefined reference to `loop'`”, wir haben keine Funktion “loop” definiert.

Interessant ist,

1. dass das in der “`function `main'`” passiert, das lässt auf ein C-Hauptprogramm schliessen.
2. dass die Datei “`main.cpp`” heisst, das bedeutet wir programmieren in C++.
3. wenn es jemand interessiert kann er sich den Code hier “`hardware/arduino/avr/cores/arduino/main.cpp`” anschauen,
4. für schwierigere Fälle kann man detailliertere Fehlermeldungen bekommen.

Wir ergänzen die loop-Funktion.

```
void setup() {
  Serial.begin(9600);
```

```
  Serial.println(LED_BUILTIN);
}
void loop() {
}
```

Wenn wir das Fenster “Serial Monitor” offen haben steht dort “13” (oder “1313”).

“LED_BUILTIN” ist also nur der Wert 13 und das Oszilloskop muss am Pin 13 angeschlossen werden.

“println“ ist eine Funktion die den Übergabeparameter “” in ASCII-Text umwandelt, mit einem Zeilenend ergänzt und zur “**Serial**” schickt.

Weil wir gerade dabei sind was ist “HIGH”, “LOW” und “OUTPUT” ? Aber das …

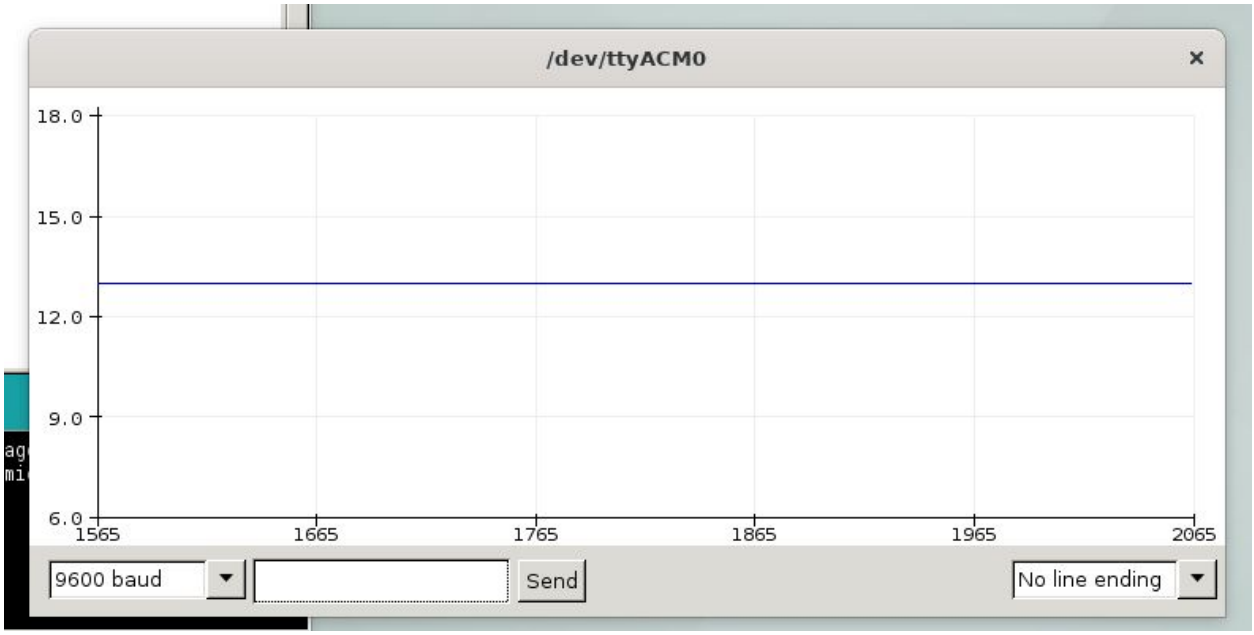
Plots

Im Menu “Tools” steht unter “Serial Monitor” “**Serial Plotter**”, Tastenkürzel Strg+Shift+L

Für einen Plot wird es Zahlen benötigen, also geben wir eine Zahl aus.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(13);
}
```

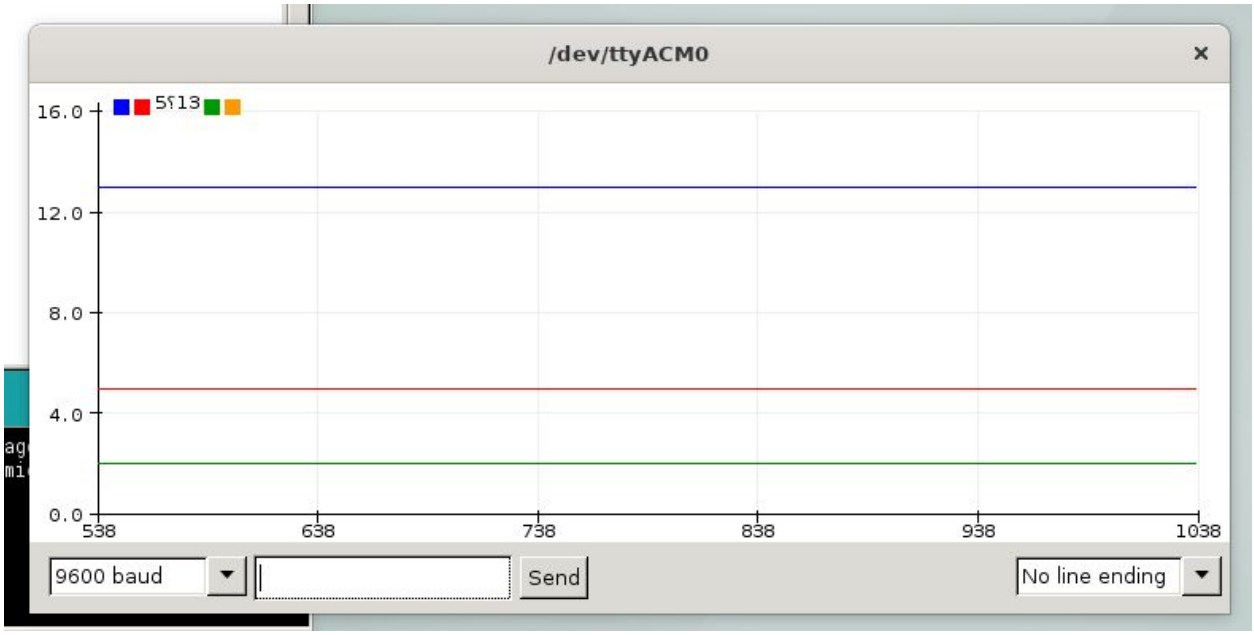
Und öffnen den Serial Plotter (Strg+Shift+L).



Ob und wie das mit mehreren Zahlen geht steht sicher in irgendeiner Dokumentation … wir versuchen einmal schnell etwas.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println("13 5 2");
}
```

Der Serial Plotter zeigt



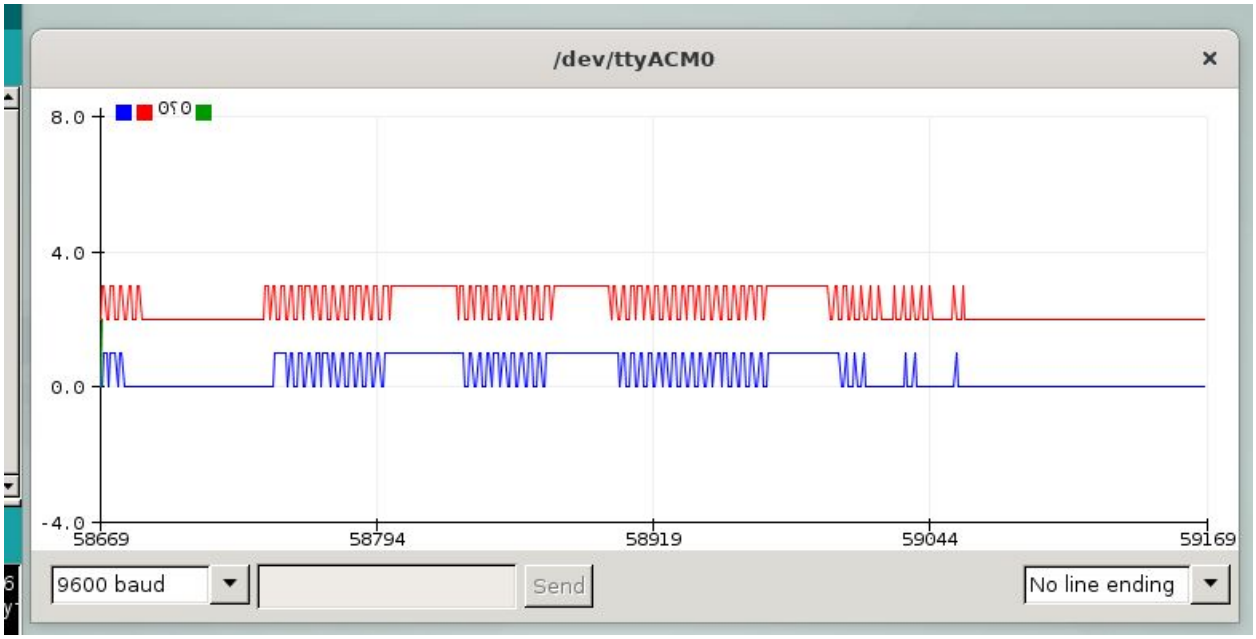
Wie die Farben zugewiesen werden und ob man auch Label vergeben kann ?

Logikanalysator

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(digitalRead(A0));
  Serial.println(" ");
  Serial.println(digitalRead(A1));
  Serial.println("");
}
```

Die Linien werden übereinander geplottet. Wir trennen die Grundlinien.

```
void loop() {
  Serial.println(digitalRead(A0));
  Serial.println(" ");
  Serial.println(digitalRead(A1)+2);
  Serial.println("");
}
```



Es fehlt (TODOs)

- Mehr Kanäle
- Kanalnamen
- Zeitspur
- Stop/Pause Funktion

Mehr Kanäle

Wir machen eine Liste (ein Array)

```
int Pins[4] = { A0, A1, A2, A3 };
void loop() {
  for (int i=0; i<4; i++) {
    Serial.println(digitalRead(Pin[i]) + i*2);
    Serial.println(" ");
  }
  Serial.println("");
}
```

"`Pin[4]`" ist ein Array mit 4 Einträgen, die wir mit "`{ A0, A1, A2, A3 }`" festlegen. In "`Pin[4]`" steht dann "`A0`" usw. Schwierig ist "`Pin[4]`" gibt es nicht, nur `Pin[0]`, `Pin[1]`, `Pin[2]` und `Pin[3]`, das sind allerdings "`4`" Einträge.

“`for (int i=0; i<4; i++) { }`” erzeugt eine Variable i, setzt diese auf den Wert 0, und erhöht (inkrementiert) i um 1 (i++) bis sie nicht mehr kleiner 4 (i<4) ist. Vor jedem inkrementieren wird der Code zwischen den geschwungenen Klammern ausgeführt. Diese Variable i ist nur zwischen den Klammern sichtbar (existiert außerhalb nicht).

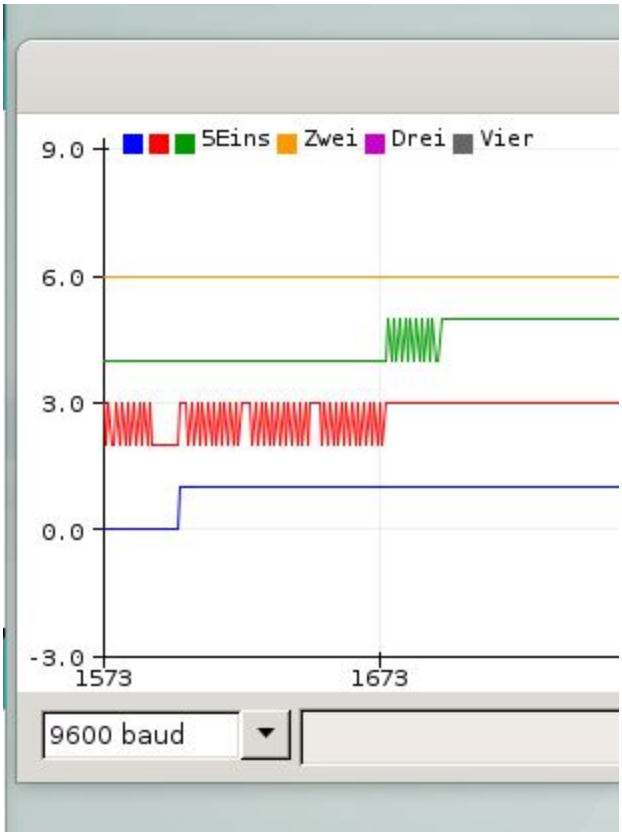
Kanalnamen

Im Plot ist oben eine Legende der Kanalfarben, dort stehen manchmal Zahlen. Wie kann man dort Kanalnamen anzeigen.

RTFM (Read The Friendly Manual) oder ausprobieren.

```
void setup() {
  Serial.begin(9600);
  Serial.println("Eins Zwei Drei Vier");
}
int Pins[4] = { A0, A1, A2, A3 };
void loop() {
  for (int i=0; i<4; i++) {
    Serial.println(digitalRead(Pin[i]) + i*2);
    Serial.println(" ");
  }
  Serial.println("");
}
```

Der Plot wird nicht richtig zurückgesetzt, man sieht noch alte Farben und Zahlen.



Zeitspur

Wir können einen zusätzlichen Kanal ausgeben, der alle 1000ms 10 ausgibt.

Stop/Pause

Die Serial-Verbindung kann nicht nur ausgeben (print, write) sondern auch einlesen.

Auf dem Serial-Monitor-Fenster ist eine Eingabezeile. Hier kann man Text eingeben und zum arduino schicken.

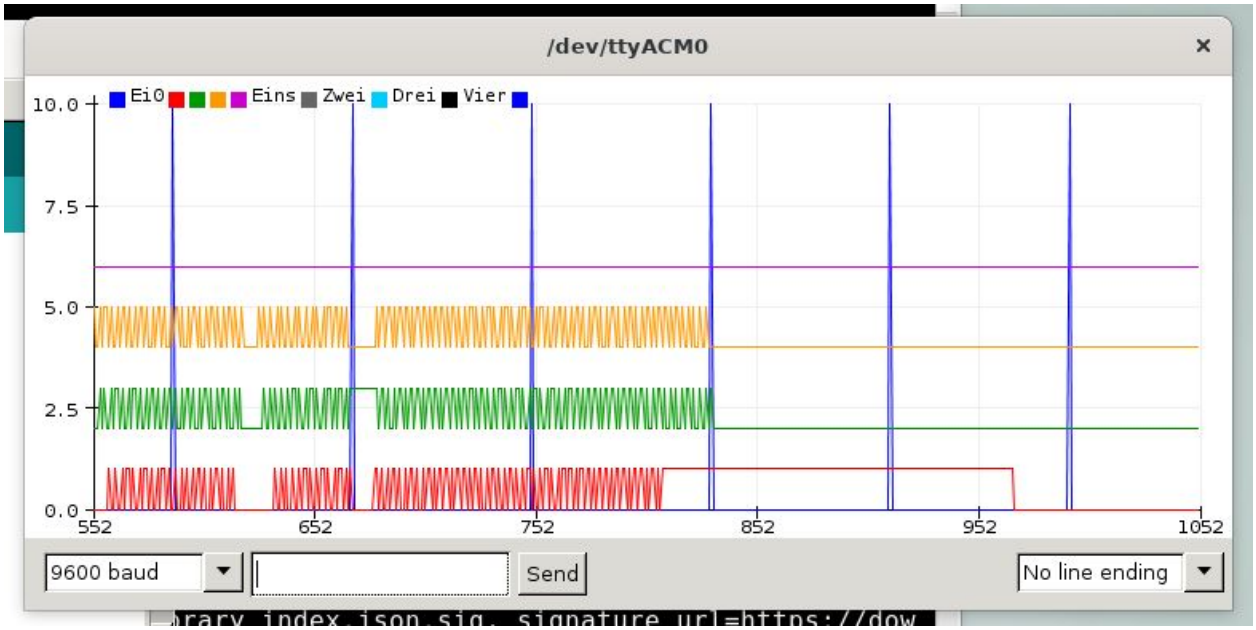
Beim Serial-Plotter gibt es das erst ab arduino 1.18.13.

Alles zusammen

... gehört noch aufgeräumt

```
void setup() {
  Serial.begin(9600);
  Serial.println("Eins Zwei Drei Vier");
}
int Pin[4] = { A0, A1, A2, A3 };
int ms = 0;
bool print_ = true;
void loop() {
  if (Serial.available()) {
    Serial.read();
    print_ = ! print_;
  }
  if (!print_) {
    return;
  }
  int t = millis();
  if ((t - ms) > 1000) {
    Serial.print(10);
    ms = t;
  }
  else {
    Serial.print(0);
  }
  Serial.print(" ");
  for (int i = 0; i < 4; i++) {
    Serial.print(digitalRead(Pin[i]) + i * 2);
    Serial.print(" ");
  }
  Serial.println("");
```

```
}
```



Dokumentieren

```
// logic analyzer

// Kanalnamen und Pins
char Names[] = "Eins Zwei Drei Vier";
int Pin[4] = { A0, A1, A2, A3 };
void setup() {
  Serial.begin(9600);
  Serial.println(Names);
}

int ms = 0;
bool pause = false;
void loop() {
  // eine Eingabe schaltet von Pause auf Run und umgekehrt.
  if (Serial.available()) {
    // die Zeile bis zum Linefeed lesen
    Serial.readStringUntil(10);
    // pause umschalten
    pause = ! pause;
```

```
  }
  if (pause) {
     return; // nichts ausgeben damit der Graph stehen bleibt
  }
  // Zeitraster: bei jedem 1000-er eine Spitze
  int t = millis();
  if ((t - ms) > 1000) {
    Serial.print(10);
    ms = t;
  }
  else {
    Serial.print(0);
  }
  Serial.print(" ");
  // Kanaele lesen und ausgeben
  for (int i = 0; i < 4; i++) {
    Serial.print(digitalRead(Pin[i]) + (i+1) * 2);
    Serial.print(" ");
  }
  Serial.println("");
}
```

- “Eingabe” meint hier wenn vom PC etwas zum arduino geschickt wird. Unterhalb des Plots ist ein “Send”-Knopf (bei arduino 1.18.13), links davon das Eingabefeld und rechts davon das Zeileende das geschickt werden soll. Das Zeilenende auf Newline stellen, dann muss man nur “Enter” drücken.
- pause umschalten: “`pause = ! pause;`”
  pause ist gleich logisch invertiert voriger Wert von pause.

Aufräumen zeitkanal

Wir extrahieren Zeitraster in eine Funktion.

```
// logic analyzer

// Kanalnamen und Pins
char Names[] = "Eins Zwei Drei Vier";
int Pin[4] = { A0, A1, A2, A3 };
void setup() {
```

```
  Serial.begin(9600);
  Serial.println(Names);
}

void print_zeitraster() {
  static int ms = 0;
  // Zeitraster: bei jedem 1000-er eine Spitze
  int t = millis();
  if ((t - ms) > 1000) {
    Serial.print(10);
    ms = t;
  }
  else {
    Serial.print(0);
  }
}

bool pause = false;
void loop() {
  // eine Eingabe schaltet von Pause auf Run und umgekehrt.
  if (Serial.available()) {
    // die Zeile bis zum Linefeed lesen
    Serial.readStringUntil(10);
    // pause umschalten
    pause = ! pause;
  }
  if (pause) {
     return; // nichts ausgeben damit der Graph stehen bleibt
  }
  print_zeitraster();
  Serial.print(" ");
  // Kanaele lesen und ausgeben
  for (int i = 0; i < 4; i++) {
    Serial.print(digitalRead(Pin[i]) + (i+1) * 2);
    Serial.print(" ");
  }
  Serial.println("");
```

```
}
```

Die Variablen “ms” und “t” werden nur innerhalb der Funktion print_zeitraster verwendet. Deshalb sollte man sie dort deklarieren. Die Variable “ms” muss aber den Wert vom letzten Aufruf enthalten, darf nicht jedesmal auf 0 gesetzt werden. Um das zu erreichen muss sie entweder ausserhalb der Funktion deklariert werden, als globale Variable, oder man schreibt “`static`” davor.

Aufräumen pause

Wir können mit der “pause” ähnlich verfahren wie mit dem Zeitraster. Diesmal muss die Variable “pause” aber in der Funktion “set_pause” und in “loop” sichtbar sein, deshalb bleibt sie global.

```
// logic analyzer

// Kanalnamen und Pins
char Names[] = "Eins Zwei Drei Vier";
int Pin[4] = { A0, A1, A2, A3 };
void setup() {
  Serial.begin(9600);
  Serial.println(Names);
}

void print_zeitraster() {
  static int ms = 0;
  // Zeitraster: bei jedem 1000-er eine Spitze
  int t = millis();
  if ((t - ms) > 1000) {
    Serial.print(10);
    ms = t;
  }
  else {
    Serial.print(0);
  }
}

bool pause = false;
void set_pause() {
```

```
  // eine Eingabe schaltet von Pause auf Run und umgekehrt.
  if (Serial.available()) {
    // die Zeile bis zum Linefeed lesen
    Serial.readStringUntil(10);
    // pause umschalten
    pause = ! pause;
  }
}
void loop() {
  set_pause();
  if (pause) {
     return; // nichts ausgeben damit der Graph stehen bleibt
  }
  print_zeitraster();
  Serial.print(" ");
  // Kanaele lesen und ausgeben
  for (int i = 0; i < 4; i++) {
    Serial.print(digitalRead(Pin[i]) + (i+1) * 2);
    Serial.print(" ");
  }
  Serial.println("");
}
```

Und dann können wir die Variable “bool pause” durch eine Funktion “bool pause()” ersetzen:

```
bool set_pause() {
  static bool _pause = false;
  // eine Eingabe schaltet von Pause auf Run und umgekehrt.
  if (Serial.available()) {
    // die Zeile bis zum Linefeed lesen
    Serial.readStringUntil(10);
    // pause umschalten
    _pause = ! _pause;
  }
  Return _pause;
}
```

Wie die nun nur noch intern sichtbare Variable “pause” heisst ist dann schon nicht mehr so wichtig … nehmen wir “_pause”. Das Programm wird nicht kürzer dadurch aber die Funktion loop wird kürzer und leicht überblickbar:

```
void loop() {
  if (pause()) {
    return; // nichts ausgeben damit der Graph stehen bleibt
  }
  print_zeitraster();
  // Kanaele lesen und ausgeben
  for (int i = 0; i < 4; i++) {
    Serial.print(digitalRead(Pin[i]) + (i + 1) * 2);
    Serial.print(" ");
  }
  Serial.println("");
}
```

Das ganze Programm (mit einem Rechteck am Pin 8).

```
// logic analyzer

// Kanalnamen und Pins
char Names[] = "Eins Zwei Drei Vier";
int Pin[4] = { A0, A1, A2, A3 };
int OutputPin = 8;
void setup() {
  Serial.begin(9600);
  Serial.println(Names);
  pinMode(OutputPin, OUTPUT);
}

void print_zeitraster() {
  static int ms = 0;
  int t = millis();
  // Zeitraster: bei jedem 1000-er eine Spitze
  if ((t - ms) > 1000) {
    Serial.print(10);
    ms = t;
```

```
  }
  else {
    Serial.print(0);
  }
  Serial.print(" ");
}

bool pause() {
  static bool _pause = false;
  // eine Eingabe schaltet von Pause auf Run und umgekehrt.
  if (Serial.available()) {
    Serial.readStringUntil(10); // die Zeile bis zum Linefeed
lesen
    _pause = ! _pause;
  }
  return _pause;
}

void loop() {
  static int loopcnt = 0;
  loopcnt++;
  digitalWrite(OutputPin, (loopcnt / 10) % 2);
  if (pause()) {
    return; // nichts ausgeben damit der Graph stehen bleibt
  }
  print_zeitraster();
  // Kanaele lesen und ausgeben
  for (int i = 0; i < 4; i++) {
    Serial.print(digitalRead(Pin[i]) + (i + 1) * 2);
    Serial.print(" ");
  }
  Serial.println("");
}
```

REFERENCES

REVISION HISTORY

- 0.2 20.11.2020 - Typografische Anführungszeichen in Programmcode korrigieren, pause, Zeitkanal, refactoring.